1 Assignment 9 (100 points)

To submit your work, collect your inputs not provided as part of the assignment, the required outputs, and your answers to the written questions in pdf format (preferred) or Word, or .txt . Clearly label all of your work. For the math-oriented portion of the assignment, you may typeset in LaTeX or Word or provide a clear photograph of a hand-written solution. Work that is not legible will not be graded. Name a zip file `CS585_Assignment9_username.zip` and submit with web-submit. There is no code to submit for this assignment.

Collect your .cpp files and label code that you changed with comments containing your username like: `//Modified by wenxinf`

1.1 Learning Objectives

- Gain familiarity with using active contours
- Review linear algebra that we will need to use when learning about camera geometry

1.2 Technical Task (points)

For this assignment, we will implement the active contours algorithm by Williams and Shah that we discussed in class (paper posted on Piazza). Code has been provided to manage user input, manage state, and compute the basic operations from the paper. You must assemble the provided pieces in order to fill in the function `evolvePoint` which will update the location of each point in the contour.

1. (Given) User-input handling to allow the user to draw an initial contour. Click to start. Drag. Click to stop. Press 'e' to start the curve evolution.
2. (Given) Various helper functions to compute basic things from the paper
3. (Given) Contour evolution function that manages the state of the contour as a whole
4. (Required) Use the helper functions provided to fill in the `evolvePoint` function. Once you have completed your implementation, you should debug it using the images I have provided (I recommend the yellow_egg.jpg image to start).

   If your first try doesn't work as you expect, try re-initializing. If, after a few tries, it is not working as you expect, you can try adjusting the regularization parameters, α, β, and γ. For me, the algorithm converges with the parameters provided in the skeleton code, so if you need to make drastic changes, you may need to double-check your implementation.

   Once your implementation is working, take a picture of two things in your environment – something simple on an easy background, and something harder with extra texture in the background or concavities. For each object, draw an initial contour, and let the algorithm run. For each object, submit 3 images of the curve evolution showing (something close to) the initial contour, some stage of the evolution in the middle, and the final contour after the algorithm has stabilized. The outline of your harder object does not need to be perfect.

1.3 Lecture Preparation (points)

Starting next week and continuing to the week thereafter, we will talk about camera geometry. This topic will build on the basic linear algebra proficiency we have developed, but we need to introduce a few more ideas before we can get started.

1. Camera Projection Matrix: When we discussed the pinhole model, we used similar triangles to compute the image coordinates of a 3D world point X so that the image coordinate $u = fX/Z$. We are now going to represent this pinhole camera projection in a different way, using a camera projection matrix. We will use this representation in many ways over the next two weeks.

   To do this, we need to remember about homogenous coordinates. In 2D, the image point (u, v) could be represented as $[u, v, 1]^T$ (a column vector). It turns out that we can do the same trick with a 3D point, so the world point (x, y, z) can be represented as $[x, y, z, 1]^T$ (also a column vector). The most important thing about homogenous coordinates is that the last element must be equal to one, so if we do some matrix multiplication where the last element ends up not equal to one, we must divide through to get back to homogenous coordinates.

   Show that performing perspective projection by multiplying a 3D point (x, y, z) in homogenous coordinates by the 3 x 4 camera projection matrix below, and then re-normalizing to obtain image coordinates $(u, v, 1)$ is equivalent to the pinhole projection model we learned in the second week of class.

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} ? \\ ? \\ 1 \end{bmatrix}$$

2. Cross product as matrix. From your linear algebra class, remember that the cross product of two 3D vectors gives you an output vector which is perpendicular to the two inputs. A very important fact to remember is that the cross product of two vectors that are parallel is the zero vector $[0, 0, 0]$. Finally, remember that the cross product does not commute and that $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$. `http://mathworld.wolfram.com/CrossProduct.html`.

   You can also represent the cross product as a matrix multiplication with a special skew-symmetric matrix. Given the vectors $\mathbf{u}$ and $\mathbf{v}$, the cross product is computed as:

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

   Verify the following identities, using the skew-symmetric matrix representation of the cross product. We will use this when deriving and exploring the properties of the Fundamental Matrix, a special matrix that describes the relationship between two cameras.

$$\mathbf{u} \times \mathbf{v} = [\mathbf{u}]_\times \mathbf{v} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$$\mathbf{u} \times \mathbf{v} = [\mathbf{v}]_{\times}^{\mathbf{T}} \mathbf{u} = \begin{bmatrix} 0 & v_3 & -v_2 \\ -v_3 & 0 & v_1 \\ v_2 & -v_1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

3. Cross Product of two vectors in homogenous coordinates to get line. Here is a neat party trick. We all learned how to find the slope and x-intercept of a line connecting two points in high school, but did you know that you can do it by just computing the cross product of two vectors? It's true! Given two image points, (m, n) and (s, t), show that if you represent those points in homogenous coordinates and take the cross product, this will give you the coefficients of the line $Ax + By + C$. (Hint: plug and chug.)

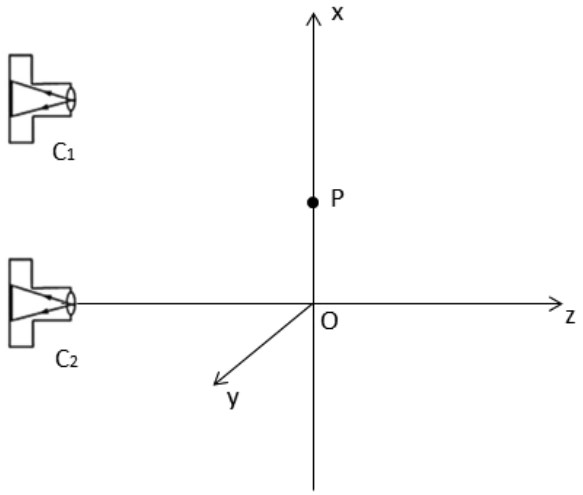
$$\begin{bmatrix} m \\ n \\ 1 \end{bmatrix} \times \begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = \begin{bmatrix} A \\ B \\ C \end{bmatrix}$$

4. Previously, when we studied the pinhole camera model, we assumed that the position of the camera was at the origin. In the picture below, the camera C_1 is in the canonical position that we are now familiar with. Assume that the world point $P = (x, y, z)$. The Y axis, shown as a diagonal line into the 3rd quadrant, is sticking straight up out of the page.

   (a) Write down the image coordinates u in terms of x, z, and the camera focal length f. Given the image below, does the point appear on the left or right side of the image?

   (b) A second camera, C_2 is translated from the origin by moving it to the left by t units. Write down the image coordinate u in terms of x, z, t and f. Given the image below, does the point appear on the left or right side of the image?

   (c) Modify the camera matrix given in the first lecture preparation question to represent the translation of the camera.



   (d) A third camera, C_3 is located at the origin, but it is rotated by $\theta = 45°$. Which side of the image will the image point be on?

(e) You will need to apply some rotation to the world point before projecting it onto the camera in order to compute the image coordinates. Assume that you can obtain a correct rotation matrix $R(\theta)$. Use your intuition to reason about what rotation matrix you should apply to the point in this case, $R(45°)$ or $R(-45°)$. Hint: What do you need to do to the point so that you can pretend that the camera is oriented in the canonical way?